

Solar roof renders that sell installs

two systems that find buildings with a bare roof, put a real solar array on that exact roof in a real photo, and send it to the person who owns it. everything here works anywhere in the world and every prompt is written out in full.

1.1 rooftop render for homes

1.2 warehouse and commercial roof render



the only made up picture in this guide. every other image is a real building, found with the searches inside and rendered from its own real photo.

SECTION 1 · 2 · 3 What this is, and how it works

you show a building owner a photograph of their own roof with solar panels already on it. that picture is the whole product.

this works because nobody ignores a picture of their own house. a solar leaflet gets binned because it is the same piece of paper as every other house on the street. a photo of *their* roof gets looked at, because it could only have been made for them.

you are not installing anything. you find the buildings, make the picture, send it, and sell the reply to a solar installer who does the work. that is the business.

every system in this seven part series is the same eight steps.

learn them once here and the other six guides are the same shape with a different picture at the end.

the one rule that keeps you out of trouble: never put a number on the picture that you cannot point at a source for. this only works because the person believes it. the moment you invent a saving figure, or round a roof size up because it sounds better, you are doing what everyone else does and you will get the reply rate everyone else gets. if you do not have a number, leave it off.

the eight steps

- 1 find** a list of buildings in one town.
- 2 qualify** each roof, and throw away the bad ones.
- 3 capture** the real photo of the good ones.
- 4 work out** the three honest numbers.
- 5 render** the panels onto that real photo.
- 6 put** the picture and the numbers on one page.
- 7 look at it yourself** and approve it.
- 8 send it.**

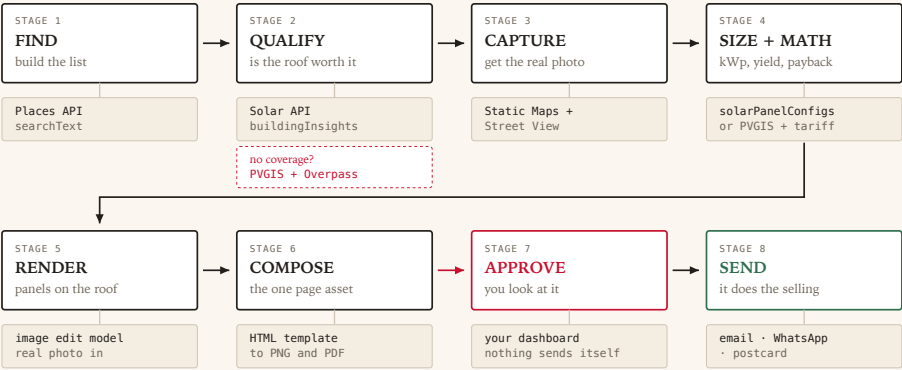
you do not need to be a developer

steps 1 to 4 are web addresses you paste into a browser. step 5 is a prompt. step 8 is an email. the only real skill is **taste in step 7**, deciding whether a picture looks believable, and you get that by looking at twenty of them.

steps 1 to 6 are automatic, step 7 is you every time, step 8 is one email. **do it by hand for one building before you automate anything.** you will understand the whole thing in an afternoon instead of spending a week building a machine for something you have never done once.

The pipeline, stage by stage

left to right, one box per stage. the data source that feeds each stage sits underneath it.



every stage writes its result back to one row in the database. a stage that fails leaves the row where it was, so the next run picks it up again and carries on from there.
stage 7 is a person, not a job. the send stage refuses to read a row that nobody approved.

Fig 1. the eight steps. the grey box under each one is where the data comes from.

SECTION 6 What you need before you start

five accounts. two of them need a card. do the Google one first, because everything waits on it and it takes the longest.

ACCOUNT	WHAT IT IS FOR	FREE TIER	CARD
Google Cloud console.cloud.google.com	one project, one api key , and you switch on five apis inside it. this single key does steps 1, 2 and 3.	10,000 free calls a month per api. the one you use most, Solar buildingInsights , is in that band, so in practice you will not pay for the roof data.	yes
switch on exactly these five: Places API (New) · Solar API · Street View Static API · Maps Static API · Geocoding API		then restrict the key to those five and nothing else, on the same screen. a leaked open key is your bill.	
an image model aistudio.google.com or replicate.com	step 5, putting the panels on the roof. this is the only part with a real cost per building.	none for images. Google's model is gemini-3-pro-image, Replicate's is openai/gpt-image-2. the pictures in this guide were made with the Replicate one.	yes
Resend resend.com	step 8, sending the email	3,000 emails a month, 100 a day, on one domain you verify. plenty.	no
Supabase supabase.com	somewhere to keep the buildings and the pictures. skip this on day one and use a spreadsheet and a folder.	500 MB database, 1 GB of files. note a free project pauses after 7 days of nobody touching it.	no
PVGIS re.jrc.ec.europa.eu	the backup for countries Google's Solar API does not cover. see the last page.	completely free. no account, no key, nothing to sign up for.	no

do this before you write a line of code

- 1 · make the Google Cloud project and switch on the five apis.
- 2 · make one api key and **restrict it** to those five apis.
- 3 · set a **budget alert** on the project. you will forget a loop is running at some point, and this is the difference between noticing today and noticing on the invoice.
- 4 · paste your own home address into the Solar API call on the next page and check the answer looks right. that one test tells you the whole chain works.

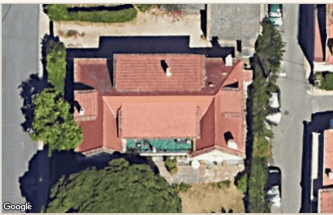
what you can skip at the start

a dashboard, a database, a scheduler, a domain, a company. none of it is needed to send the first ten pictures.

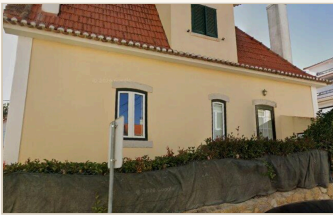
a spreadsheet with one row per building, a folder of before and after images, and your normal email account will take you all the way to a paying client. build the app when doing it by hand starts to hurt, and not before. section 10 is there for when that day comes.

a homeowner sees a photo of their own house with solar panels on it before anyone has knocked on their door.

SECTION 4 · 5 What it does, and who pays you for it



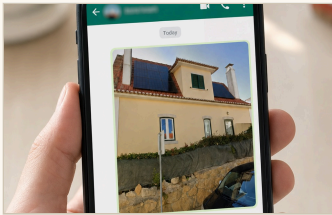
1 · google measures the roof



2 · you pull their real photo



3 · the model puts panels on it



4 · it lands on their phone

who pays you. a solar **installation** company. not a manufacturer, not a national brand. the small firm that turns up in a van and fits the panels. two to about twenty people is the sweet spot: below that nobody answers the phone, above it they have a marketing person and you become a line item.

they pay because an installer with idle fitters loses money every day, and a lead that arrives with a picture converts faster than one that arrives as a name.

what they do instead today: buy shared leads from a comparison site, where the same homeowner is sold to four installers at once and only the fastest caller wins. they complain about it constantly, which makes it an easy conversation to start.

the beginner's version of this, and honestly the one to do first

do not contact homeowners at the start. a house has no listed email address, so you would be stuck with physical post before you have proved anything.

instead, run steps 1 to 6 on **thirty houses in one installer's town** and email **the installer** the thirty pictures. installers are businesses, so they have an email address you can actually find. you are selling them a ready made campaign instead of hunting down thirty strangers. it is faster, it is easier, and it gets you a paying client sooner.

SECTION 7 The pipeline, step by step

every example below is one real house: Rua Dr. António Martins, Estoril, Portugal, at 38.7043566, -9.393056. the answers shown are the real answers. paste them and you will get the same thing.

1 Find the buildings

pick one suburb, not a city. get the addresses on a few streets and turn each into a latitude and longitude with the Geocoding API. a spreadsheet of thirty is plenty to start.

2 Ask Google about the roof

one call per building. this is the important one and it is the whole system.

```
curl "https://solar.googleapis.com/v1\
/buildingInsights:findClosest\
?location.latitude=38.7043566\
&location.longitude=-9.393056\
&requiredQuality=HIGH\
&key=YOUR_KEY"
```

what comes back for this house: a **240 m² roof**, room for **61 panels**, **1,853 hours of sun a year**, and a list of every roof slope with the direction it faces. the best slope here faces 182°, which is almost dead south.

if you get a 404, that is not a bug. it means Google has no roof data for that country. do not crash, use the PVGIS backup on the last page.

4 Work out the three numbers

the answer from step 2 has a list of options. **do not take the biggest one.** 61 panels means covering the north facing slopes too, and the picture will look wrong to anyone who knows roofs. take the one that fits on the best slope. for this house that is 24 panels.

the three numbers are:

24 panels	from the api's own list
9.6 kWp	24 × 400 watts ÷ 1000
16,255 kWh a year	the api's own figure

never turn kWh into money. you do not know their electricity tariff and you never will. state the production and let the installer talk about money. this also keeps you honest and out of trouble.

5 Put the panels on the photo

send the real photo and the prompt on the next page to an image model. that is the whole step.

if it comes back wrong, just run it again before changing anything. these models are random and a plain retry fixes a lot of bad ones.

6 One page: picture on top, numbers underneath

big picture, then the three numbers small, then one line saying where the measurement came from and that the layout is an illustration. that line is what makes it credible instead of spam, so do not shrink it to nothing.

3 Get the real photo

two calls, and **the first one is free and does not use any quota**, so always do it first.

```
# is there a photo here at all?
curl "https://maps.googleapis.com/maps/api\
/streetview/metadata?location=38.7043566\
,-9.393056&radius=60&key=YOUR_KEY"

# if that said OK, get the picture
curl -o before.jpg "https://maps.googleapis\
.com/maps/api/streetview?size=640x640\
&location=38.7043566,-9.393056\
&heading=293&pitch=12&fov=70&key=YOUR_KEY"
```

heading is the compass direction the camera looks in. get it wrong and you photograph the house opposite. work it out from the camera position that the first call gives you to the building, or just try a few numbers by hand at the start. **pitch 12** tilts the camera up to see over hedges, which you will need on about one house in four.

7 Look at it yourself

one question: **would i believe this if it arrived about my house?** at the start you will bin about one in four. that is normal and it is why this step is a person. the most likely way this whole thing goes wrong is you getting bored and approving without looking.

8 Send it

picture in the body of the email, four lines of text around it, and one easy question at the end. the prompt for that text is on the next page.

SECTION 9 The picture, and what makes it work

a real one, made exactly the way the last page describes. Rua Dr. António Martins, Estoril, Portugal. real Street View photo, real Solar API numbers, real model output.



Before. straight from Street View, free. heading 293, pitch 12 to see over the hedge.



After. one pass, no touching up. 24 panels. the panel shadows fall the same way as the chimney shadow, and the hedge, wall, car and sky are untouched. **this is the one you send.**

the same house, done badly

this is also real model output from the same photo. i asked for it deliberately, because every mistake a beginner makes is in this one picture, and once you have seen it you will spot all five in a second.



Wrong. bright uniform blue with a visible grid. runs edge to edge with no gap at the eaves or the ridge. panels overhang into thin air on the right. the chimney has been wrapped in panels. and the light is coming from the wrong side of the sky.

five rules, and this picture breaks every one

- 1 • always start from their real photo.** never generate a house from a description. the whole effect is them recognising their own gate.
- 2 • match the light already in the picture.** the most common giveaway. check where the chimney shadow falls, then where the panel shadows fall. if they disagree, bin it and run it again.
- 3 • leave a gap at the edges, and never cover a chimney.** real arrays stop short of the roof edge, because of wind rules and because fitters need somewhere to stand. edge to edge is what installers spot first, and a chimney painted over is obviously fake to the person who lives there.
- 4 • the number in the picture and the number in the text must match.** if the text says 24 panels and the picture clearly shows 40, you have lost them.
- 5 • write "illustration, not a design" under it, and keep the before photo forever.** the line costs you nothing, because they are already looking at their own roof by the time they read it.

WHAT GOES UNDER THE PICTURE, WORD FOR WORD

24 panels • 9.6 kWp • roughly 16,255 kWh a year.

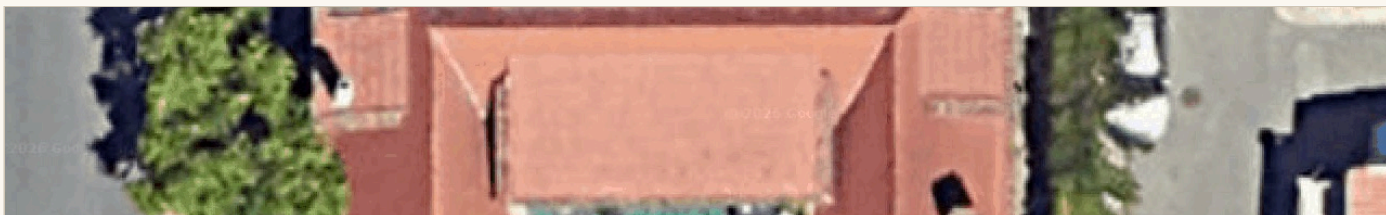
roof measured by the Google Solar API from aerial imagery dated 16 December 2024. panel layout is an illustration, not an installation design.

the thing nobody tells you about these models

they do not paint onto your photo. they make a **new photo** that looks like yours. the house, the light and the framing survive, but small details do not always: do the same house from above and you will see a parked car quietly change shape. that is fine for what this does, and it is not fine if you ever call it a photograph. say "**your roof, with an array on it**". never say "a photo of your roof".

SECTION 8 The prompts

three prompts, complete, nothing to fill in except the `{{double braces}}`. these are the guide. everything else you could work out, but these took a lot of failed pictures to get right.



What the first prompt is deciding about. Google reports every one of these slopes separately with its angle and its compass direction. the prompt below picks the one facing the sun, checks it is big enough and not too steep, and throws the building away if it is not worth your time.

STEP 2 · DECIDE IF THE ROOF IS WORTH IT · ANY CHEAP TEXT MODEL. PASTE THE API ANSWER IN AT THE BOTTOM.

You are screening roofs for a solar sales picture. You will be given the raw JSON answer from the Google Solar API buildingInsights endpoint for one building. Decide whether this roof is worth making a picture for.

Apply these rules in order and stop at the first one that fails:

1. REJECT if solarPotential.maxArrayPanelsCount is less than 8.
Reason: "too small to be worth an installer's time".
2. REJECT if no entry in solarPotential.roofSegmentStats has stats.areaMeters2 of at least 20.
Reason: "no single roof slope big enough for a tidy array".
3. Find the BEST SLOPE: of the slopes with areaMeters2 of 20 or more, pick the one whose azimuthDegrees is closest to `{{EQUATOR_AZIMUTH}}`.
(`{{EQUATOR_AZIMUTH}}` is 180 if the building is north of the equator, 0 if it is south of it.)
4. REJECT if that best slope's azimuthDegrees is more than 90 degrees away from `{{EQUATOR_AZIMUTH}}`, measured the short way round the circle.
Reason: "best roof slope faces away from the sun".
5. REJECT if that best slope's pitchDegrees is below 5 or above 45.
Reason: "roof angle outside the straightforward install range".
6. REJECT if imageryDate is earlier than `{{MIN_IMAGERY_YEAR}}`.
Reason: "photo too old to trust that the roof is still bare".

If nothing rejects it, ACCEPT.

Then choose how many panels to draw. From solarPotential.solarPanelConfigs, pick the LARGEST option whose panelsCount, multiplied by (panelHeightMeters * panelWidthMeters), is still no more than 80% of the best slope's areaMeters2. Never pick the biggest option in the list. If nothing qualifies, pick the smallest option.

Return ONLY this JSON object and nothing else. No explanation, no code fences.

```
{
  "decision": "accept" | "reject",
  "reason": "one short lowercase sentence",
  "best_slope": { "pitch_degrees": number, "azimuth_degrees": number, "area_m2": number },
  "chosen_panel_count": number,
  "system_kwp": number,
  "yearly_kwh": number,
  "confidence": "high" | "medium" | "low"
}
```

How to work out the numbers: system_kwp is chosen_panel_count times solarPotential.panelCapacityWatts divided by 1000, rounded to one decimal place. yearly_kwh is the yearlyEnergyDcKwh of the option you chose, rounded to a whole number. Never invent a number that is not in the JSON you were given. Set confidence to "low" if imageryQuality is not HIGH.

Here is the building:

```
{{BUILDING_INSIGHTS_JSON}}
```

STEP 5 · PUT THE PANELS ON THE HOUSE · GEMINI-3-PRO-IMAGE OR GPT-IMAGE-2. SEND THE REAL PHOTO WITH IT.

Photorealistic edit of this real street photograph of a house at {{ADDRESS}}. Add a real rooftop solar photovoltaic array onto the roof slopes that face the camera.

Use {{PANEL_COUNT}} standard all-black monocrystalline panels, each about {{PANEL_H}} m by {{PANEL_W}} m, laid in neat landscape rows flush and parallel to the existing roof pitch, mounted on low-profile black rails about 10 cm above the {{ROOF_MATERIAL}}, with a consistent gap from the ridge and from every roof edge, so no panel overhangs the eaves or the hips.

The panels must follow the exact perspective, vanishing lines and pitch of the real roof planes in this photo. Give the panel glass a slightly reflective dark blue-black finish that picks up the same sky and the same light already in the photograph, and cast soft panel shadows onto the roof in the same direction as every other shadow in the scene. Leave a clear walkway strip around the chimney and do not cover the chimney, the skylights or the roof vents.

Change nothing else at all. Keep the walls, the windows, the doors, the garden, the boundary wall, any parked vehicles, the sky, the exposure, the grain and the framing exactly as they are.

The result must look like an unretouched photograph of a house that already has solar panels, not like a rendering or a diagram. No text, no labels, no watermarks, no arrows, no overlays.

STEP 5, COMMERCIAL VERSION · SEND THE SATELLITE TILE WITH IT, NOT A STREET PHOTO

Photorealistic edit of this real top down satellite image of {{BUSINESS_TYPE}} at {{ADDRESS}}. Cover the large {{ROOF_DESCRIPTION}} in the centre of the roof of the frame with a commercial rooftop solar photovoltaic array.

Use dense rows of dark blue-black framed panels laid flat and flush with the existing roof bays, running parallel to the long axis of the building, in continuous blocks separated by clear walkway gangways about one metre wide every few rows, so a maintenance crew can walk the roof.

Keep a clear perimeter setback of about one and a half metres from every roof edge and around every existing rooflight strip, vent and plant unit already visible on the roof. Do not cover {{EXCLUSIONS}}.

Match the exact top down camera angle, the resolution, the satellite softness, the lighting and the colour balance of this image. Panels should read as a real installed array seen from a satellite, with subtle variation in reflection across the roof, not as a flat blue rectangle.

Change nothing else in the frame. The car park, the parked vehicles, the delivery yard, any stacked materials, the trees, the roads and every neighbouring building stay exactly as they are.

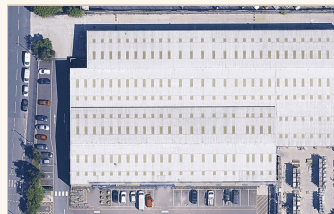
No text, no labels, no outlines, no highlight boxes, no arrows, no watermarks.



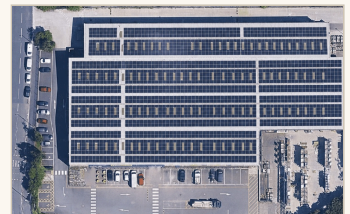
in · the real street photo



out · the house prompt



in · the satellite tile



out · the commercial prompt

filling in the braces, and the one that matters most

{{ROOF_MATERIAL}} · look at the photo and write it. "tiles", "slate", "corrugated sheet".

{{BUSINESS_TYPE}} · "a large builders merchant warehouse", "a garden centre". stops the model inventing a building.

{{ROOF_DESCRIPTION}} · what you can actually see: "white sawtooth warehouse roof". **this is the substitution that most improves the picture.**

{{EXCLUSIONS}} · the bits that must stay clear: "the entrance canopy along the bottom".

never say "the south facing side"

the model has no idea which way is north. describe the roof slope by **where it is in the picture**: "the slope facing the bottom of this image", "the left hand slope". say "south facing" and it will guess, and it guesses wrong about half the time. this one mistake causes more bad pictures than everything else put together.

Write the message that goes with a solar picture being sent to the owner of a specific building. The picture does the selling. Your job is the few lines around it that make opening it feel normal rather than strange.

Facts you may use, and you may not use anything else:

- address: {{ADDRESS}}
- panels that fit: {{PANEL_COUNT}}
- system size: {{SYSTEM_KWP}} kWp
- estimated yearly production: {{YEARLY_KWH}} kWh
- where the roof measurement came from: {{DATA_SOURCE}}
- date of the aerial photo it was measured from: {{IMAGERY_DATE}}
- who is sending it: {{SENDER_NAME}} at {{SENDER_COMPANY}}

Rules:

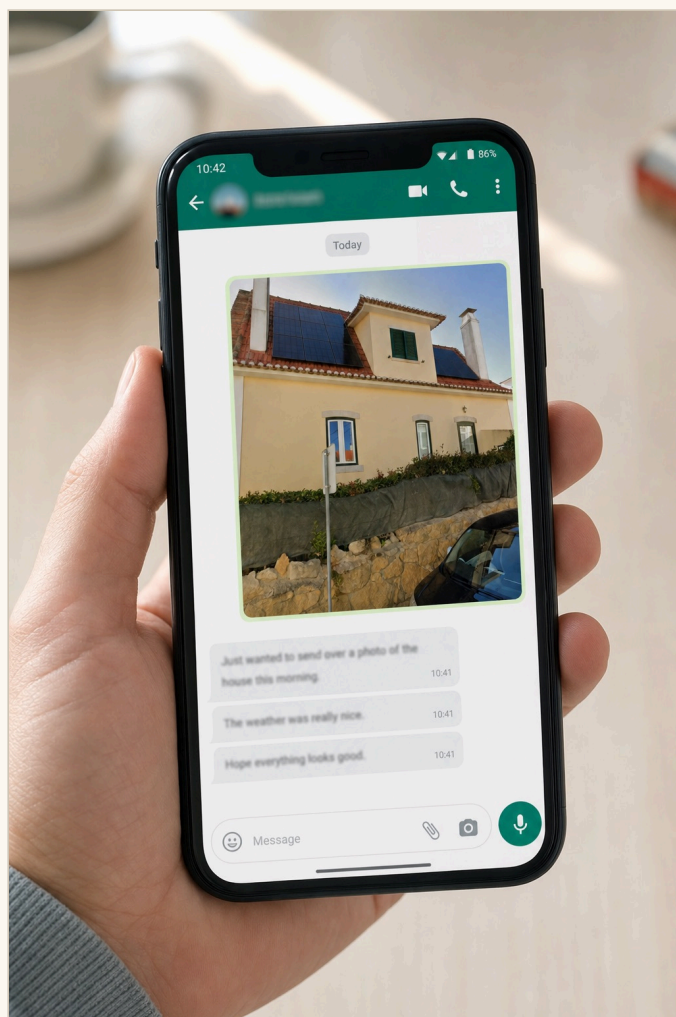
- Under 60 words.
- All lowercase is fine and preferred. Plain everyday words.
- No em dashes. Use full stops and commas.
- Open by saying plainly what the picture is. Do not open with a greeting formula, a compliment, or a question about their energy bills.
- State the production figure once. Never turn it into money, a saving, a percentage or a payback period.
- Say where the roof measurement came from, in four words or fewer.
- Say clearly and briefly that the picture is an illustration, not a design.
- End with one easy question that can be answered in one word.
- Do not use: unlock, transform, revolutionise, game changer, exciting, solution, reach out, touch base, circle back, I hope this finds you well, act now, limited time, exclusive, no obligation.
- Do not claim you have visited the property, surveyed it, or been asked to look at it. Do not imply you already know them.

Return only the message. If it is an email, return a JSON object with keys "subject" and "body", where the subject is under 50 characters and does not contain the words free, save, or offer.

what that gives you for the Estoril house, on whatsapp:

this is your roof at rua dr. antónio martins with panels on it. google's aerial survey says 24 panels fit, about 9.6 kWp, roughly 16,255 kWh a year. the picture is an illustration, not a design. worth a proper look?

forty three words. the picture does the work, so the words only have to make opening it feel normal. no greeting, no compliment, no question about their energy bills, and no money figure anywhere.



Roughly what lands. that is the real Estoril render sitting inside the message. the wording in this mockup is placeholder, not the copy on the left.

warehouse and commercial roof render

the same eight steps pointed at buildings with five thousand square metres of empty roof, and a company that owns it.

SECTIONS 4 · 5 · 6 · 7 What it does, who pays, and what changes

build this one first if you only build one. warehouses are *businesses*, so Google lists them with a name, an address, a website and a phone number, which solves the contact problem that makes the houses version awkward. their roofs are flat and empty, so the picture comes out better. and the arrays are twenty to a hundred times bigger, so one deal is worth a lot of houses.

who pays you. a commercial solar installer, five to fifty people. what they do instead is bid on tenders, competing on price against strangers. getting in front of an owner *before* a tender exists is worth more than any number of leads inside one.

why it lands. most business owners have no idea how much roof they have, so five thousand square metres is new information about their own property, and that is hard to ignore. the roof is also dead space that costs money and earns nothing.

the four things that change

WHAT	THE CHANGE
step 1 gets easy	use the Places API text search instead of hunting addresses. one search per town gives you the building <i>and</i> the contact details.
step 3 uses satellite, not Street View	the top down view is the only angle that shows how much roof there is. zoom 19 for a small site, zoom 18 for a big one, always scale=2.
step 2 thresholds go up	in the filter prompt: 800 m² of roof and 250 panels minimum, max pitch 15, and delete rule 4 entirely , because flat roof panels sit on tilted frames and face wherever you point them. leaving that rule in throws away your best buildings. present the count rounded down to the nearest 50.
step 8 adds one sentence	they are open during the day, so most of what the roof makes gets used on site instead of exported cheaply. that one sentence is the commercial argument.



And this is why you look. Longacres Garden Centre, Ashford. the Solar API reports 6,111 m² and room for 2,129 panels, and it is right about the area. most of that roof is **glass**, which cannot carry panels, and no field in the response tells you so. same problem with a roof that already has an array on it. both are obvious in the tile and invisible in the data.

the searches that actually find big roofs

do not search "warehouse". search the kinds of business that sit in big single storey buildings and tend to own them, one town at a time:

```
curl -X POST 'https://places.googleapis.com/\n/v1/places:searchText' \n-H 'Content-Type: application/json' \n-H "X-Goog-API-Key: YOUR_KEY" \n-H 'X-Goog-FieldMask: places.id,\nplaces.displayName,places.formattedAddress,\nplaces.location' \n-d '{"textQuery":"builders merchant in Bristol",\n"pageSize":20}'
```

swap the search for: garden centre · cold storage (the best, because refrigeration runs all day) · furniture warehouse showroom · distribution centre · printing works · joinery.

four of those searches across three English towns found, among others: Selco Bristol 5,073 m² / 2,362 panels, Longacres Ashford 6,111 m² / 2,129, JD Furniture Birmingham 6,354 m² / 2,528. real answers. about a third of a good search is big enough to bother with.

what it looks like when it lands

real, end to end. **Selco Builders Warehouse, Cater Road, Bishopsworth, Bristol BS13 7UH**, found with the search above. 5,073 m² of bare roof, room for about 2,350 panels, roughly 940 kWp. imagery dated 31 July 2024.



Before. satellite tile, zoom 19, centred on the roof rather than the Places pin, which usually sits on the entrance or the car park.



Right. rows follow the existing roof bays, walkways run between the blocks, the rooflight strips still read through, and the yard is untouched. **the walkways are what make it look installed rather than generated.**



Wrong, and this is the usual failure. one flat slab of blue. no rows, no gangways, no gap at the roof edge, the rooflights painted over, and it spills onto the tarmac. an installer needs one second to dismiss this.

the two things you must check by eye, every time

does the roof already have panels? far more common on commercial than houses, and no api will tell you. sending someone panels on a roof that already has them ends the relationship instantly. it is right there in the tile. look.

is it glass? garden centres return huge roof areas and much of it is glasshouse, which cannot take panels. the api will happily say 2,129 panels fit. again, look.

SECTIONS 10 · 11 · 12

When you outgrow the spreadsheet

only build this once doing it by hand hurts, which is usually when a client starts paying monthly. until then a spreadsheet, a folder of pictures and your normal email account is genuinely the right answer. skipping ahead to this page is the most common way people never send anything.

three tables. **prospects** is one row per building with its roof numbers. **assets** is the before picture, the after picture and the message, with a status that starts at pending. **sends** is one row every time something goes out. you do not have to write the sql yourself: install the Supabase CLI and paste the prompt below into an AI coding tool.

then one scheduled job, weekday mornings, so the queue is waiting for you when you start work rather than filling up while you sleep. OpenClaw is the easiest way to set it up, but a plain cron job calling a url does the same thing. three things to get right, and they are the only three:

1 • one job at a time. if two copies run at once you will send the same person two pictures.

2 • retry three times, then stop. wait a minute, then five, then twenty five, then park it and tell you. a job that retries forever quietly burns api calls on a broken key.

3 · the approval gate is one column, not a feature. the status on the asset starts at pending, and the sending code only ever reads rows that say approved. build it that way and no bug and no confused agent can ever send something you did not look at.

The three screens you need

anything beyond these three is you avoiding the part where you send it.

1 · /prospects

the list

all · queued · rendered · waiting · sent · dead

Selco Builders Warehouse, Bristol
5,073 m2 roof · 2,362 panels

waiting

Rua Dr. Antonio Martins, Estoril
240 m2 roof · 61 panels max

sent

Longacres Garden Centre, Ashford
6,111 m2 roof · 2,129 panels

dead

sorted by roof size, biggest first. the biggest roof is always the call you make first.

one click on a row opens screen 2.

2 · /prospects/[id]

preview and approve

BEFORE

the real photo

AFTER

the render

9.6 kWp · 24 panels · 16,255 kWh a year
imagery Dec 2024 · Google Solar API · HIGH

the message, editable, right here
so you can fix one word without a redeploy

approve and send

reject

redo render

3 · /sends

the log

when	who	channel	state
09:04	Selco Bristol	email	delivered
09:04	Longacres Ashford	email	bounced
09:05	Estoril villa	whatsapp	read
09:06	Hows Gray Bristol	postcard	in transit
09:06	Cousins Birmingham	email	REPLIED

this screen is the only proof the system works.
a bounce is a dead address, not a dead prospect.
fix the address and send it again.

one row per send, ever. never overwrite one, or
you lose the only record of what you actually said.

The three screens, and nothing else. the numbers shown are the real Solar API figures for the real buildings in this guide. anything you want to add beyond these three is a way of avoiding the part where you send things to strangers.

SET UP THE DATABASE · PASTE INTO CLAUDE CODE, CURSOR, OR ANY AI CODING TOOL, IN YOUR PROJECT FOLDER

```
Set up Supabase for this project using the Supabase CLI, and write the SQL migration for me.

First, if it is not already set up: install the CLI, run `supabase init`, then `supabase link` to my
project, and tell me exactly what to paste when it asks for my project ref and database password.

Then create a migration with three tables. Use uuid primary keys with gen_random_uuid() as the
default, and give every table a created_at that defaults to now().

TABLE 1, prospects. One row per building I am considering.
- owner_id, text, not null. This is my Clerk user id.
- system, text, not null. Either 'residential' or 'commercial'.
- name, text, nullable. The business name. Null for a house.
- address, text, not null. lat and lng, both float8, not null.
- place_id, text, unique, nullable. The Google Places id. Null for a house.
- roof_area_m2, panel_count, system_kwp, yearly_kwh. The numbers from the Google Solar API.
- imagery_date, date. When the aerial photo the measurement came from was taken.
- raw, jsonb. The entire Solar API response, kept as is. Never throw this away.
- status, text, not null, default 'new'.

TABLE 2, assets. One row per picture I make for a prospect.
- prospect_id, references prospects, and delete it if the prospect is deleted.
- before_url and render_url, text. Where the two pictures live in storage.
- subject and body, text. The message that goes with it.
- model, text, and prompt, text. Which image model made it and the exact prompt used, so that when
the pictures get worse in three months I can prove what changed.
- status, text, not null, default 'pending'. Only ever 'pending', 'approved' or 'rejected'.

TABLE 3, sends. One row every single time something goes out. Never updated, only inserted.
- asset_id, references assets, and delete it if the asset is deleted.
- channel, text. One of 'email', 'whatsapp', 'postcard'.
- to_address, text, not null. provider_id, text, the id the provider gives back.
- status, text, not null, default 'queued'. sent_at, timestamptz, nullable.
```

Now the security, and this part matters most. Turn on row level security on all three tables, then write policies so I can only ever see and change my own rows. prospects has owner_id on it directly, so compare that against the Clerk user id, which Supabase exposes as auth.jwt() -> 'sub'. assets and sends have no owner column of their own, so their policies have to join back up to prospects to work out who the row belongs to.

Also create a private storage bucket called 'assets' for the pictures, with a policy that only lets me read files inside a folder named after my own user id.

Finally, add the indexes that actually matter for how I will use this. I will be listing prospects by owner and status, sorting them by roof area, looking up assets that are still pending, and looking up sends by the provider id when a delivery webhook comes back.

Explain in plain English what each policy does before you write it, run the migration with `supabase db push`, then show me one test query proving a second user cannot read my rows.

LEAD GEN SYSTEMS · GUIDE 1 · SOLAR

11

the two ways to charge

run it yourself. you keep the code and the keys, and everything goes out branded as the client, from their domain, with their name on it. the homeowner never learns you exist, which is the point: they are being contacted by a solar installer about their roof, which is normal, rather than by an agency, which is not. you bill monthly on the leads or the booked appointments.

or hand it over. you build it in their accounts, teach someone to press approve, and charge a setup fee plus a monthly. the monthly is not optional: this leans on four outside services and one of them changes every few months.

which fits this? run it yourself. someone has to look at every picture and judge whether it is believable, and that takes taste you only get from seeing a few hundred. an installer handed this approves everything on the first morning, sends a picture of a roof that already has panels, and decides the system does not work.

with a first client, sell a pilot, not a monthly

one town, an agreed number of pictures, two weeks, fixed fee. you do not know your reply rate in their market yet and neither do they, so a monthly agreed up front is a guess by both of you. at the end you both have real numbers and you price off something measured. clients say yes to this far more easily.

before you send the first one

- [] the Google key is **restricted** to the five apis, and there is a budget alert on the project
- [] you ran the Solar API on your own home and the answer looked right
- [] you have made **at least twenty pictures** and looked at every one, so you know what a bad one looks like before a stranger does
- [] your sending domain has SPF, DKIM and DMARC set up and tested
- [] the picture says "illustration, not a design" and where the measurement came from
- [] a reply lands in an inbox somebody actually reads

what breaks

the roof already has panels. the most embarrassing one, and no api warns you. you catch it by looking at the tile. forever.

the pictures quietly get worse. image models change under you without warning. save the exact prompt and model with every picture so you can prove what changed, then retune the prompt.

the Places pin is not the building. it lands on the entrance or the car park. recentre on the building position the Solar API gives you before pulling any image.

what i have not tested

everything up to and including the picture was run against the live apis on the two real buildings in here. i have **not** sent a real one to a real prospect, stood up the dashboard, or run the scheduled job. those come from current documentation. and i have **no measured reply rate** for this, in any country. anyone who gives you one is guessing. run the pilot and measure your own.

if you are not in Europe or north america

Google's Solar API covers **over 472 million buildings in 40+ countries**. an experimental setting adds ~150 million more, including Brazil, Colombia and Peru, by adding `&requiredQuality=BASE&experiments=EXPANDED_CO` to the call. outside that it returns **404**, which is an answer, not an error.

when it does, get the roof outline free from OpenStreetMap at `overpass-api.de`, estimate panels at $\text{roof area} \div 1.96 \text{ m}^2$ then take 70%, and get the yearly kWh free from **PVGIS**, which needs no account at all.

and change the channel. in brazil, mexico, most of latin america, india, indonesia, nigeria, kenya and south africa, **whatsapp is the business channel**, not an alternative to email. marketing there needs an opt in, so use it as the *reply* channel after a first contact elsewhere.



The other way it lands. the same Selco render printed and posted. a house has an address and usually no findable email, so post is the honest channel for residential, and almost nobody else is using it. print.one and PostGrid both mail worldwide from an api.